EXHIBIT
1100
10·8·2013
PENGAD 800-631-6989

Chimera 3.2

A real-time operating system developed at CMU which is far better than anything else you'll find on the market.

Table of Contents

Overview
Examples
Cool Stuff
References
See Also
VASC Contact
History

Overview

The Chimera 3.2 Real-Time Operating System has its own HTML page which more than adequately covers what it does, and how to use it. Let me therefore take this opportunity to answer the question, "Why use Chimera instead of VxWorks?" (Besides the fact that it has a wonderful graphical interface called Onika , I mean.)

Here's what Chimera inventor David B. Stewart has to say on the subject:

Frequently Asked Questions about Chimera vs. VxWorks

Q: What are the similarities and differences between the VxWorks and Chimera real-time kernels?

A: As with most other commercial real-time operating systems, VxWorks only provides a basic real-time kernel which gives you task management, low-overhead context switching and local semaphores, static highest priority first scheduling, and an interface to program the hardware timers for periodic events.

Chimera provides an extended real-time kernel. In addition to all the basic functions that VxWorks provides, it also gives you both static and dynamic real-time scheduling, virtual timers which eliminate the need to explicitly program hardware timers, the novel deadline failure handling and global error handling mechanisms, and two-level device driver support.

Q: Chimera claims it is a "multiprocessor operating system", while VxWo
"multiprocessor support". What is the difference?

A: VxWorks is only a single-processor operating system with multiprocess

ABB v ROY-G-BIV
Trial IPR2013-00063
ABB EXHIBIT 1100

EXHIBIT
1100
10·8·2013
PENGAD 800-631-6989

Chimera 3.2

A real-time operating system developed at CMU which is far better than anything else you'll find on the market.

Table of Contents

Overview
Examples
Cool Stuff
References
See Also
VASC Contact
History

Overview

The Chimera 3.2 Real-Time Operating System has its own HTML page which more than adequately covers what it does, and how to use it. Let me therefore take this opportunity to answer the question, "Why use Chimera instead of VxWorks?" (Besides the fact that it has a wonderful graphical interface called Onika , I mean.)

Here's what Chimera inventor David B. Stewart has to say on the subject:

Frequently Asked Questions about Chimera vs. VxWorks

Q: What are the similarities and differences between the VxWorks and Chimera real-time kernels?

A: As with most other commercial real-time operating systems, VxWorks only provides a basic real-time kernel which gives you task management, low-overhead context switching and local semaphores, static highest priority first scheduling, and an interface to program the hardware timers for periodic events.

Chimera provides an extended real-time kernel. In addition to all the basic functions that VxWorks provides, it also gives you both static and dynamic real-time scheduling, virtual timers which eliminate the need to explicitly program hardware timers, the novel deadline failure handling and global error handling mechanisms, and two-level device driver support.

Q: Chimera claims it is a "multiprocessor operating system", while VxWorks only claims it has "multiprocessor support". What is the difference?

A: VxWorks is only a single-processor operating system with multiprocessor extensions. Those

extensions involve networking multiple VxWorks systems over the backplane using the TCP/IP protocol. The only difference is that the communication medium is the backplane, and not ethernet. This setup is no more a multiprocessor setup than two Sun workstations on the same local area network.

Chimera is a multiprocessor operating system because the kernels on multiple Real-Time Processing Units (RTPUs -- i.e. single board computers) can communicate with each other via Chimera's high performance system-level communication, called express mail. Without such kernel-level interprocessor communication, an operating system cannot make interprocessor communication transparent to the user's programs running on each RTPU, and limit both the types of higher-level communication mechanisms that can be designed and their performance.

Q: We have written a communications layer above VxWorks? Will this work with Chimera?

A: Because Chimera is a multiprocessor operating system, there is no need for such a communications layer. Almost anyone using multiple processors and VxWorks is required to write such a layer, because of VxWorks' lack of multiprocessor support. In Chimera, all the communication mechanisms are built-in to the operating system, so you don't have to write them. Because they are built-in, everyone uses the same ones; as opposed to what happens today with projects using VxWorks, where every project or institution has their own custom layer, thus limiting any potential software reuse or technology transfer.

Q: What are the communication mechanisms available in Chimera, and how does each compare to VxWorks?

A: Here is a quick rundown of most of them:

- Express Mail: this is a high-performance real-time non-blocking communication mechnism between the kernels on each RTPU and the host workstation. This mechanism allows the other interprocessor communication and synchronization mechanism to operate transparently across multiple processors.

- Global Shared Memory: tasks can dynamically create and attach to shared memory on any processor, and access that shared memory. Because of the inter-kernel communication in Chimera, all the VMEbus address offsets can be calculated automatically during the initialization of the shared memory, and all operations are transparent to which processor the segment is actually on. In VxWorks, addresses of shared memory segments must be hard-coded into user's code, with the offsets (with are RTPU dependent) also added manually. This makes the code non-portable, and difficult to maintain if the hardware configurations are changed, as each hardware change requires a modification of the source code. The only reason this is available in VxWorks is because the VMEbus allows it; i.e. VxWorks makes no special provisions for global shared memory. In Chimera, no such modifications are required, and the same executable can be run even if the base addresses of any of the RTPUs in the system are changed.

- Spin Locks: This is a front end to the lowest-level of interprocessor communication using the atomic test-and-set (TAS) instructions. VxWorks lets you use the TAS instructions, but do not provide a convenient front-end for using them and automatically timing out if the lock cannot be obtained.

- Remote Semaphores: In addition to local semaphores, Chimera provides remote semaphores which allow tasks on all RTPUs to use a semaphore. VxWorks has only local semaphores; interrupts must explicitly be sent by the user if multiprocessor synchronization is required.

- Prioritized Message Passing: Chimera's message passing is multiprocessor. Any task on any RTPU can send or receive a message to/from any queue. Messages can be retrieved either first-in-first-out, last-in-first-out, or highest-priority-first. In VxWorks, message passing is local to an RTPU. To send a message to a remote RTPU, either sockets must be used, or a communications layer must implement the message passing on top of shared memory with interrupts for signaling. It is not clear from VxWorks literature, but queues might only be first-in-first-out.

- Global State Variable Table: Chimera provides this communication mechanism for predictable real-time communication and making efficient use of the VMEbus bandwidth. One global table exists, and every task which attaches to the table has a local copy of the parts of the table it needs. Updates of the local and global tables are done periodically and predictably. This is a 'must' communication mechanism for developing reconfigurable software. VxWorks does not provide anything comparable.

- Subsystem Multiprocessor Task Control: Chimera allows a task on one RTPU to control tasks (i.e. spawn, block, etc.) on multiple RTPUs. This is another feature required when developing reconfigurable software which must execute on multiple processors. VxWorks does not provide anything comparable.

- Triple Buffer Communication Mechanism: This predictable real-time communication mechanism allows Chimera to communicate with other subsystems which may or may not be running Chimera, or to communicate with intelligent memory mapped I/O devices or special purpose processors. The code for the mechanism is extremely portable, so that it can be compiled to execute on any non-Chimera platform. VxWorks does not provide any such mechanisms.

- Host Workstation Interface: Chimera allows processes on the host workstation to attach to the real-time environment by appearing to the RTPUs as just another RTPU. This allows communication between the host and RTPUs to occur in the same transparent manner that RTPUs communicate with each other. For example, a graphics application on the host can display real-time data by reading from a shared memory segment on one of the RTPUs. In VxWorks, any host-to-RTPU communication must be done through the ethernet, which is slow, less predictable, and much more difficult to program.

- Transparent Host Procedure Calls: RTPUs can execute remote procedure calls on the host workstation, either to execute some non-real-time code or to obtain information that is otherwise not available on the RTPUs. For example, RTPUs have a physical clock, but they don't have a time-of-day clock. So calling the function 'gettimeofday()' is automatically transfered into a remote procedure call, and the time-of-day from the host is returned. An RTPU can also execute programs like 'ls', 'emacs', etc. These programs "appear" to run on the RTPU, but in fact execute on the host workstation, but the results are sent to the RTPU's 'stdout', and input is received from the RTPU's 'stdin'.

- Special Purpose Processors (SPP): Chimera treats special purpose processors, such as floating point

accelerators, LISP machines, and image processors, as slaves, which execute a short, efficient, non-preemptive executive. A programming running under Chimera can then execute a procedure remotely. The SPP drivers automatically handle the communication and synchronization between the RTPU and the SPP. VxWorks has no equivalent interfaces.

- Interrupts: Chimera provides a hardware indepedent interface for generating and handling both VMEbus and mailbox interrupts. Chimera also gives you 256 mailbox interrupts per RTPU, even if the RTPU only supports one or two mailbox interrupts (as is the case with most RTPUs). VxWorks does not provide any special support for interrupts; you are at the mercy of whatever the hardware gives you, and must program the hardware using non-portable code.

Q: How does the hardware support compare between Chimera and VxWorks?

Chimera has concentrated on providing as much functionality as possible along a single line of RTPUs, initially. However nothing precludes poviding support for multiple RTPU's. We expect that this will happen in due time. At present, though, VxWorks supports several different models of RTPUs.

Chimera provides several device drivers for off-the-shelf I/O hardware, such as DAC, ADC, parallel I/O and serial I/O. VxWorks does not provide such drivers.

Chimera also provides device drivers for some special purpose processors, allowing user's to access the SPP's through the Chimera generic SPP interface. VxWorks does not have any such drivers. Several SPP hardware vendors do sell VxWorks drivers; however, those drivers always have custom interfaces, thus making them non-portable with the interfaces of other SPPs.

Q: How does the technical support for Chimera software compare to the support given for VxWorks by Wind River Systems?

Chimera II did not have any official support; however, those using Chimera II can usually get an answer or bug fixes within a couple of days directly from the authors. With Chimera 3.0, similar support will be available from the authors. However, when the software is commercialized, it will be possible to buy a software support contract, in which case there will be official support.

Q: What is the performance of Chimera vs. other RTOS?

A: The biggest strength of Chimera is in all the added features which are not available in other real-time operating systems (RTOS). However, despite the additional functionality, Chimera's performance is at par with VxWorks and other RTOS. Some benchmarks for real-time kernels were published comparing a few commercial RTOS (A. Topper, "A computing architecture for a multiple robot controller, M.S. Thesis, McGill University). Those same benchmarks were executed on Chimera, following are the results:

[NOTE: The authors are providing these results without any explicit or implied warranties.]

Test name	pSOS+	*BOS	LynxOS	VxWorks	Chimera
Create Delete Task	58	21	--	142	42
Ping Suspend Resume	14	16		4	11
Ping Semaphore	18	21	31	32	43
Get/Release Semaphore	11	55	4	11	13

Queue Fill	46	26	140	20*	66^
Queue Drain	43	29	132	22*	61^
Queue Fill/Drain	91	59	278	44*	120^
Queue Fill, Urgent	47	27	170	72*	67^
Alternate Queue F/D	238	252	867	371*	570^
Allocate Memory	40	27	57	68	42
Deallocate Memory	38	33	20	83	36

* VxWorks does not support message passing. These times are from simulating message passing through the use of shared memory. Note: the times above are as quoted in the thesis. This must have been an older version of VxWorks, since VxWorks now has a message passing mechanism described in their sales brochures. The numbers given in the sales brochure are the following:

- msgQSend
  - no tasks pending: 77
  - task pending: 129
  - queue full: 68
- msgQReceive
  - message available: 73
  - message unavailable: 65

Based on these numbers, we would expect the Queue Fill to be 77, Queue Drain to be 73, Queue Fill/Drain to be 150, and Alternate Queue F/D to be at least 450. It is unclear whether "Queue Fill, Urgent" is possible, as the sales brochure does not specify if messages can have priorities.

^ Chimera's message passing is a multiprocessor mechanism. Chimera's kernel does not have built-in message passing. The timings are for local messages when using the multiprocessor message passing mechanism in the Chimera communication library.

Q: With VxWorks, an RTPU can communicate directly with a different computer system on the network. Can I do anything like that in Chimera, considering you don't yet support ethernet in the backplane?

A: Yes, you can. Chimera allows communication by an RTPU over ethernet with other systems by using the host workstation's ethernet. An easy to use interface is a front end to the awkward socket() interface required to send and receive messages over ethernet. Going through the host workstation is completely transparent to the programmer. With this setup, you have not lost any performance as compared to VxWorks, which does support an ethernet in the backplane: in VxWorks, one RTPU still has to communicate to the RTPU hosting the ethernet. However, the Chimera design remains consistent with the global/local operating system decomposition, where the local operating system performs all the real-time stuff, and the global operating system takes care of the non-real-time or soft-real-time stuff like the file system, workstation graphics, and networking.

Q: Chimera 3.0 is advertised as a real-time operating system for "reconfigurable sensor-based control systems". Why is it preferred to use Chimera over VxWorks for reconfigurable systems?

A: In order for a software module to be reconfigurable, it must satisfy the two following conditions: 1- the module is independent of target hardware 2- the module is independent of target application

Because multiprocessor operations on VxWorks are not transparent, and VxWorks does not provide any generic interfaces to special purpose processors or sensors and actuators, the first condition

above is violated.

All Chimera multiprocessor communication is hardware independent. Chimera also has two-level device driver support for I/O devices and sensor-actuator interfaces, and a generic SPP interface, providing additional hardware independence to non-general-purpose-CPU hardware.

Chimera has many tools built-in to the operating system to support the reconfigurable software framework being developed at CMU. The state variable table mechanism, configuration file reading utility, and subsystem task control mechanisms are but a sample of those tools, which simplify the task of writing reconfigurable software modules which are independent of the target application. Since VxWorks does not have any such tools, it is easier to satisfy the second condition using Chimera than it is when using VxWorks.

Q: Our managers say that we should use VxWorks because it would keep us compatible with other institutions and other projects. Why should we break this trend and use Chimera?

A: Unfortunately this is a myth that far too many people believe. Just because multiple projects/centers use the same operating system does not mean that they will have *any* code that is reusable or compatible between them. It is a standard software architecture, and not the operating system, that provides compatibility between multiple projects. Since VxWorks does not provide support for any specific software architecture, every project uses its own architecture, and hence despite using the same operating system, the applications are not at all compatible.

Here is an example of a similar situation. Consider two projects, each using Sun SPARCstations with SunOS 4.0. One of them uses SunView, while the other is uses X windows. You develop a graphics application for X windows. It is NOT compatible with the other project using SunView, even though both are running under the same operating system. However, a third project is using a DEC PMAX, running Ultrix 3.2, and X windows. The graphics application DOES run on the DEC, even though it is using a completely different hardware architecture and a completely different operating system. The reason is they are both using the X11R5 standard protocol, which is a standard software architecture for window applications.

When developing applications under Chimera, we recommend that the reconfigurable software framework be used, so that multiple applications at multiple institutions are compatible with each other. If one project uses the framework, and the other doesn't, then you won't have compatibility between those projects. The advantage of Chimera over VxWorks in this case is that Chimera gives you all the tools necessary to easily and quickly implement an application based on the reconfigurable software framework. VxWorks, on the other hand, has no corresponding software architecture, and no corresponding tools. Every project must then develop their own architecture and tools, and hence application development takes a lot more time, and the application is not compatible with the applications in other projects or centers.

In summary, maintaining consistency between multiple project is NOT a function of the operating system; it is a function of the software architecture.

Examples

For visual examples, check out the Onika HTML page. If you want to use Chimera, here's what you need to do to your paths and environment:

```
setenv CHIMERA_ARCH $ARCH
setenv CHIMERA_DIR ~chimera/chim_3.2
setenv CHIMERA_CONFIG $CHIMERA_DIR/config/chimera.config.$HOST
setenv CHIMERA_LOCAL your_top_directory_of_Chimera_software
setpath PATH -i $CHIMERA_DIR/$CHIMERA_ARCH/bin
```

Then it's just a matter of entering (on a machine which runs Chimera -- remember, only one one user at a time can run Chimera on any given machine):

% chim

Cool Stuff

I really can't mention them all here; it's best for you to check out the Chimera 3.2 Real-Time Operating System page There's a huge manual on-line which gives programming examples, function definitions, and much more.

References

The Onika User's Manual, the Chimera 3.2 User's Manual.

See Also

Chimera, VxWorks, Onika.

History

10/14/94 Created by Matt Gertz

This page has been accessed 947 times.